

Pixel Testbeam 2004 Offline Code Instruction

Jianchun Wang
(Revised: 10/11/2004)

-
1. [A quick start](#)
 2. [The coordinate systems](#)
 3. [How to create your own working directory?](#)
 4. [How to check out the whole package?](#)
 5. [Brief introduction of the package](#)
 6. [Offline analysis control file](#)
 7. [Geometry configuration files](#)
 8. [Detector calibration files](#)
 9. [Normal analysis of testbeam data](#)
 10. [Dump low level event information and event display](#)
 11. [Manual alignment](#)
 12. [Automatic alignment](#)
 13. [Monte Carlo generation](#)
 14. [Further analysis using CWN ntuple](#)
 15. [Histogram IDs](#)
-

This note provides instructions on offline code of pixel testbeam 2004.

A Quick Start

Most user would not be patient enough to read through the whole note. This section is just to give you a quick start to run a data file and take a quick peep. Please follow each step and let me know if it does not work for you. More details are in following sections.

1. Login to `fnppd.fnal.gov`.
2. Change your default to a proper location where a working directory will be created.
3. Create a working directory, generate all necessary files and links and cd enter the working directory

```
/prj/btev/pixel_tb04/testbeam/scripts/anal_data 1981 tmp_1981  
cd tmp_1981
```

4. Run the analysis program

```
./monitor.exe
```

5. Start PAW, execute `manip.kumac` to view histograms. Note that you need to hit `Enter` to proceed during viewing of plots.

```
paw  
exec manip run01981
```

The Coordinate System

Before getting into the code I want to define the coordinate system first. Two different types of coordinate systems are used throughout the analysis: world coordinate system and local coordinate system of each plane. The world (lab) coordinate system is defined as that the beam along `+Z` direction, `+Y` points up, and `+X` points to the left of beam, or further into the hut. The local `+Z` direction is the normal incident direction to the detector plane that points roughly in the world `+Z` direction. Local `+Y` direction is the pixel row or column direction depending on which one points up. To convert between local coordinates and world coordinates, three angles: `alpha`, `beta`, and `phi` are used. The angle `beta` corresponds to the tilt angle of the plane that we often talked about.

How to create your own working directory?

The analysis package is checked out code on `fnppd` computer. The path is `/prj/btev/pixel_tb04`. Data, geometry, ADC calibration, ntuple/histogram and log files are either located or linked in `DATA`, `GEOM`, `CALIB`, `HIST`, and `LOG` subdirectories respectively. Source code, macro files, example of analysis are in `testbeam` subdirectory. In this note sometime the path `/prj/btev/pixel_tb04` is skipped when a relative path is provided.

Unless you need to modify the code or to install on a different computer, most users do not need to check out the whole analysis package. The users are encouraged to create their own directories, copy or link necessary files. For those who are not familiar with UNIX commands, the command to link a directory or a file is

```
ln -s source_file target_file
```

I also wrote a script to create a basic working directory for you. To study run 1981, for example, you need to

```
/prj/btev/pixel_tb04/testbeam/scripts/anal_data 1981 tmp_1981
cd tmp_1981
./monitor.exe
```

The script `anal_data` creates `tmp_1981` under current directory. It copies or links basic files including the geometry file `tb_geometry.dat` specific for the run 1981, and the executable file `monitor.exe` that analysis data file. You can analysis data of different run. In that case you may need to modify the geometry file or check `GEOM/run_geom_list.txt` and copy the proper that is already aligned.

Please note that some of the files are indeed links to a public file. Computer may allow you to overwrite them. So if you want to modify something you might want to copy instead of link to the file. Please also try not to create your own file that in the package directories.

How to check out the whole package?

If you want to modify the source code or install on a different computer you need to checkout the package. The BTeV environment parameters should already be set in BTeV user accounts. If they are not properly set the following lines need to be added in your startup files, `.cshrc` or `.tcshrc` for `cs`/`tcsh` shell users.

```
source /usr/local/etc/fermi.cshrc
source ~bphys/setup.csh dev
setenv CVSROOT ":pserver:Your_UID@btevsrv2.fnal.gov:/home3/btev/btevcvs/cvs"
```

For `sh`/`bash` users, the following command should be included in `.shrc` or `.bashrc`.

```
./usr/local/etc/fermi.shrc
. ~bphys/setup.sh dev
export CVSROOT=":pserver:Your_UID@btevsrv2.fnal.gov:/home3/btev/btevcvs/cvs"
```

Proper cvs login is also required either through `.cvspass` file or `cvs login`.

For the first time you check out code, proper directory tree needs to be created. The following commands will do the trick. The first three commands check out codes. The last two commands are of `SoftRelTool` that will take care of include files and other stuff.

```
newrel -t dev tb04
cd tb04
addpkg -t -h testbeam
srt_super_init
srt_setup -aS
```

If it is not the first time to check out the code, you might want to update the code in `tb04` directory or subdirectories by using

```
cvs update
```

Each time you have a fresh login you need to set proper environment parameters by

```
srt_setup -a
```

Brief introduction of the package

Under `testbeam` directory the interesting files that the user might need to deal with are in `scripts`, `monitor/src`, `include/monitor`, and `example` subdirectories.

The `scripts` directory contains scripts that are used to transfer data file, create working directory analysis data, generate summary plots and so on.

File	Brief explanation
<code>cp_2_fnppd</code>	Script to transfer data file from pixel07 to fnppd temporary FTP location. It is called by DAQ program once a data run is finished. It can also be interactively run by user on pixel07.
<code>cp_at_fnppd</code>	Script to transfer data file from FTP location to storage disks. It checks if the file is already copied or not from <code>DATA/FILE.log</code> . The storage disk is linked to <code>DATA/data_at_work</code> .
<code>crontab_jc</code>	Input file to crontab that starts <code>cp_at_fnppd</code> every 10 minutes.
<code>anal_data</code>	It generates a temporary working directory for a run, analyzes data and generates summary plots. It can also generate a user working directory that is similar to <code>example</code> .
<code>generate_nml</code>	It generates <code>tb_monitor.nml</code> file that is controls analysis. It is called by <code>anal_data</code> .
<code>KUMAC</code>	Directory that houses PAW macro files, same as in <code>example</code> directory.
<code>MISC</code>	Directory that houses other miscellaneous files, same as in <code>example</code> directory.

The `monitor/src` directory contains analysis source codes, either in FORTRAN or in C. The command to generate executable program is `gmake`. The data flow will be explained later in separate sections. Here is a list of some of the interesting files.

File	Brief explanation
<code>monitor.F</code>	The main program that controls everything.
<code>read_namelist.F</code>	Read in the namelist control list. The default input file is <code>tb_monitor.nml</code> which will be explained later.
<code>testbeam_dio.c</code>	Read/write binary data file, encode/decode event data.
<code>user_trigger.F</code>	Provides trigger using hit information.
<code>sixy_cluster_find.F</code>	Form clustered hits from pixel hits.
<code>sixy_cluster_position.F</code>	Determine the hit position.
<code>form_tracks.F</code>	Form a rough track using hits from each plane in an path window.
<code>kalman_fit.F</code>	Fit each rough track with Kalman filter.
<code>fill_ntuple.F</code>	Book and fill CWN for selected tracks.
<code>sixy_charge_generate.F</code>	MC generation of charge signal.
<code>mn_write_track.F</code>	Select and write track summary for auto alignment.
<code>mn_fit_geo.F</code>	Interface to auto alignment using selected tracks.
<code>dump.F</code>	Event dump.
<code>user.F</code>	User interface at various stage.

Files in `include/monitor` are header/include files for the source code. User is advised not to change the file in this directory unless it is really necessary.

The directory `example` is the place where user normally works. It is recommended, however, that you create your own working directory with script `testbeam/scripts/anal_data` as explained before. All supporting files are either copied or linked in this directory. The commonly used files are listed below with brief explanations.

File	Brief explanation
monitor.exe	This is a link to main executive program that analyze event data.
tb_monitor.nml	Main user interface file that controls how <code>monitor.exe</code> analyzes data.
tb_geometry.dat	Geometry configuration of all planes.
adc.kumac	A PAW macro file that is used to generate dummy calibration file.
adc_plane*.dat adc_fpix*.dat?	Detector calibration files.
align.kumac	A PAW macro file that is used to manually align the telescope.
manip.kumac	A PAW macro file used to view histograms.
util.kumac	PAW macro file that contains supporting macros for other macro file.
anal.f	FORTRAN function used in further analysis by <code>anal_example.kumac</code> .
anal_example.kumac	A PAW macro file to perform further analysis using CWN ntuple.
dump.dat	Output file of dump action, especially useful for event display.
mn_geometry.dat	The file of new geometry parameters after auto-alignment
mn_skip.dat	The stereo control file on parameters to fit in auto-alignment
BEAM_1980.dat	Data files taken with telescope, MC simulation, and intermediate track summary file for auto-alignment.
mcd00900.dat	
trk01980.dat	
mcd00900.hbk	Histogram/ntuple files from simulation, event analysis, auto-alignment.
mcd00900.hbk, run01980.hbk	
aln01980.hbk	
aln01980.hbk	
mcd00900.ps	Summary postscript files of MC generation, event analysis, auto-alignment generated with <code>manip.kumac</code> , and manual-alignment with <code>align.kumac</code> .
mcd00900.ps, run01980.ps	
aln01980.ps	
align.ps	

Offline analysis control file

The offline analysis control file name is either defined by environment parameter `NAMELIST_FILE`, or default `tb.monitor.nml`. It uses three namelists `CONTROL`, `FILES`, and `DETECTOR`. The user defined and default parameter values control how the program runs. The most commonly used parameters are listed below. Some of them are useful only for certain purpose selected by `tb_control`.

Parameters	Explanation
max_evts	Number of events to study, 0 means all events.
nskip	First nskip events will be skipped in analyze.
ncycle	Non-zero value results in a dump action.
dump_opt	0: Dump will prompt for commands; non-zero: It dumps automatically into a file.
dx_cuto	The X window that is used in forming a track.
rand_trg_rate	For MC simulation with random trigger event only.
xref1o,xref2o, yref1o,yref2o	The reference planes used to form track.
drop_plane(7)	Skip the plane in study even if it presents(0:not skip, 1:skip).
dut_plane(7)	The plane is the detector under test(0:no, 1:yes).
run_mc	Assign a run ID in MC simulation.
max_miss_planes	Maximum allowed missing planes including telescope planes and DUT planes.
max_miss_teles	Maximum telescope planes that can be missing.

tb_control	'event': analyze of beamtest data; 'summary': analyze data and save track for auto-align; 'align': auto alignment; 'mcgen': MC generation; 'mcran': MC generation with random trigger event; 'filter': Generate skim data file using selection in <code>user_trigger.F</code> .
geom_file	The file that contains geometry configuration.
data_in_file(1)	Input files that contains data you want to analyze or the random trigger event in 'mcran' case.
data_out_file	Override the auto-generated output file name.
hbk_out_file	Overrider the auto-generated histogram file name.
dump_file	The file that used to dump information into.
adc_file(1)	ADC calibration file for plane 1. If other planes are not defined they use that of plane 1.
sixy_ort_r(1)	The orientation of row. +1 means the larger row ID corresponds to larger X/Y values.
sixy_ort_c(1)	The orientation of column.

The geometry configuration file

The geometry configuration file is defined by `geom_file` in control file, or default `tb_geometry.dat`. The type of detector and plane IDs are also defined in this file. Thus the user should check it before run analysis code. Otherwise the output may not be what you expected.

Some parameters are kept for historical reason, or for Kalman filter purpose that user does not need to touch. The interesting parameters are explained below. The unit of length is `cm`. A "!" at the beginning of the line comments out it.

Parameters	Explanation
TYPE	SPD: pixel, SSD: strip detector; X/Y indicates the precision measurement direction.
ID	The plane ID, which does not have to be continuous or in an order. They will be resorted according to Z for Kalman filter.
xc yc zc	The approximate center location of the plane.
delx dely delz	Adjustment of center location determined by alignment.
alpha	The angle of rotation in the xy plane.
beta	The angle of rotation in the xz plane.
phi	The angle of rotation in the xy plane.

Detector calibration files

The FPIX detectors are calibrated using injected pulse. The noise level, threshold and ADC - charge response are stored in the calibration files. These files are used in reconstruction. The name of file to be used for each plane is defined by `adc_file(ip)` in the control file, where `ip` is the plane ID. For plane 1 the default file name is `adc_plane1.dat`. For other planes, if the file is not defined, the file of plane 1 will be used.

The calibration file uses free number format. The unit of charge is Ke. The format of the FPIX1 calibration files are as follows.

Line	Sample values	Explanation
1	FPIX1	Type of detector electronics.
2	1	Plane ID. If it is not correct, a warning will be sent.
	18	Number of columns
	160	Number of rows
	4	Number of parameters that characterize ADC
3	0	Column ID
	0	Row ID
	2.00 4.31 9.28 20.00	Four threshold levels of 2-bit ADC that presents ADC-charge response.
	0.20	Noise level of the electronics
...	-	Increment of Row ID and then Column ID. The order is checked with the IDs.

FPIX2 has 3-bit ADC thus it has 8 thresholds. FPIX0 has 8-bit ADC. Its ADC-charge response is characterized by a threshold and a 6-parameter function, thus in total 7 parameters.

In example directory three dummy calibration files are generated by `adc.kumac` When there is no proper calibration file, these file can be used. The penalty is that the charge weighting or eta correction is not precise.

Normal analysis of testbeam data

Before a normal analysis or a simple peep of the testbeam data, user needs to check the following:

1. `tb_control='event'` in control file.
2. `data_in_file` is the data file that needs to be studied.
3. The types of detectors in geometry file are correct and the location is not way off.
4. The detector orientations is correct (`sixy_ort_r`, `sixy_ort_c`).
5. The reference planes are the best choice.

In example directory run the program by `./monitor.exe`. At the end of running a short summary of the data will be given. It includes: total number of events analyzed, number of tracks formed, and efficiency of each plane. User needs to pay attention to `pixel hits / event` before the DAQ system is stable. If the value is way below 1, either the electronics is not working properly or the plane is away from the center of the beamline. If `pixel hits / event` of each plane is acceptable, one should check `cluster hits used / track`. It is an indication on whether the detector location in geometry file is way off or not. Note that the default value of X window in forming a track is 400 μ m, which can be enlarged before the telescope is manually aligned.

The analysis program also generate a list of histograms and a CWN ntuple. The file name is generated automatically if it is not defined by `hbk_out_file` in the control file. User is recommended not to define it. The auto-generated file name is `runxxxxx.hbk` for testbeam data or `mcxxxxx.hbk` for MC generated data, where `xxxxx` is the 5-digit run ID.

User can view the histograms in `PAW`, `ROOT` or any other compatible packages. A PAW macro file, `manip.kumac` is provided to view selected histograms. It can distinguish among normal data histogram file, MC data histogram file, MC generation histogram file and auto alignment output histogram files, and thus treat them differently accordingly. A postscript file will be generated after the view. Here is the how to run it: start `PAW` and a `HIGZ` window will pop out, run `exec manip your_histogram_file` and hit return carriage. Please note that `pto` (please turn over) option is switched on thus user needs to hit return carriage (in `HIGZ` window if you did something else after start) to view next page.

A sample of `postscript` file is generated using MC simulation data. Some histogram IDs and ranges are generated dynamically for different configuration. The number of histograms shown varies, and thus the page number differs. Here are explain the histogram just based on this file.

- Page 1 shows track information at generator level. Only 1 track per event is generated. The track position is in world coordinates at the incidence to the upper-most detector plane.
- Page 2 shows the global hit information. Note that detector planes are sorted according to Z location. That is to say the most upper-stream plane has ID (`iz=1`).
- Pages 3 to 8 show detailed hit information 1 page per plane, in the order of `iz`. The real plane IDs (eg `SPD_X 02`) are shown in histogram axis labels.
- Page 9 shows the global cluster information. One track hit on SPD plane can produce signals on more than one pixel but in connected region and forms a cluster.

- Pages 10 to 21 show detailed cluster information 2 pages per plane. A hit cluster could span over more than one pixel in small dimension thus a charge weighting and eta correction can be done. This will be done in further study since it is **beta** angle dependent. In the normal analysis only binary weighting is done.
- Page 22 shows global information of track formation. The top plot shows how many tracks had included hits from each plane. Bottom one is just a record keeper of which planes are used as reference planes.
- Page 23 shows the correlation of the reference planes. With a proper manual alignment the distributions of X or Y difference should center at 0.
- Pages 24 and 25 show the deviations of the best hit on each plane to the expectation defined by reference planes.
- Page 26 shows the global Kalman filter information. Note that the last plot is not flat due to that the resolution assigned is not optimized.
- Pages to 38 shows the Kalman filter information 2 pages per plane, with 1 page for that the hit of this plane is included in the fit, and another page excluded from the fit. For resolution measurement one needs to use "excluded" one.

Dump low level event information and event display

Sometimes one may need to access low level event information either interactively or dump directly to a file. The parameters to control are **ncycle** and **dump_opt**. The **ncycle** controls the frequency, that is, the dump happens one event out of each **ncycle** events. The **dump_opt** controls what kind of information to dump, if the value is zero then interactive mode is prompted. The options are:

Option ID	Option string	Function
1	help	list all options
2	run	continue run with next event
3	exit	continue run and stop dump
4	stop	stop the run
5	hits	list all pixel hits
6	clusters	list all cluster hits
7	tracks	reconstructed track information
8	mcgen	generated track information
9	skip	change number of events skipped
10	ncycle	change dump cycle
11	hreset	reset histogram
12	hprint	print one histogram
13	planes	list planes
14	block	dump the raw block
15	dump	dump information to a file, mainly to eventDisplay

Some options can take one or two parameters, which is taken as the first and the last elements to be shown. By default, all elements are shown.

The last option **dump** enables output of event information in ASCII format. It feeds event display program **Orsay** by Dario. User should not change the output format of this option unless it is discussed with me and Dario.

Manual alignment

The location of detector planes may not be right in the geometry file. The user needs to provide it with reasonable precision before further study. One way is not estimate using plane efficiency or dump information. A more precise way is to use **align.kumac** in PAW.

The manual alignment macro file **align.kumac** uses histograms generated by normal analysis. It first align the two X reference planes and two Y reference plane before align any other planes. To start the manual alignment you need to issue command in PAW:

```
exec align your_histogram_file 0
```

With option **0** the alignment will align X and Y reference planes first. If X and Y planes are not aligned the macro stops for

you to correct X and Y first as other planes depend on them. In cases you want to proceed, either because the reference planes are not to be aligned or because the alignment has to be performed interactively, option 1 (default) has to be used. The X and Y reference planes, however, are always checked.

The first page shows two plots of X reference planes. And the X values are converted into world coordinates. If the two planes are aligned the top scatter plot should have a bar along red line, and the bottom distribution should center at 0. The distribution is fit to a Gaussian function. The fit may not work that well since the background level can be very high. In that case user should enter `quit` to quit and fit interactively.

The offset of the peak is determined from the fit. If the offset is very small the alignment ignores and proceeds to the next step. If the offset is big enough the alignment gives suggested new geometry values in the format that use can be cut and pasted into the file. If it is fit interactively, user can change the parameter of the second reference plane directly:

```
delx(new) = delx(old) - mean.
```

The alignment of non-reference plane is done with comparison between the plane and interpolation/extrapolation of the reference planes.

Automatic alignment

The manual alignment could align the location of detector planes in quite reasonable precision. Other parameters especially for those correlated ones are not as easy as a fit to 1-D histograms. The offline analysis program, however, provides a more sophisticated fit that employs MINUIT package. The geometry parameters are varied and the best parameter values are determined with minimum chi2 of all track candidates.

The CPU time is a big issue in automatic alignment. The whole process is split in two separated steps: The raw data is analyzed and raw track candidates are formed and saved in a file; The raw track information is read and MINUIT fit is performed for selected parameters. It may take several iterations before an ideal set of parameters is determined.

The running at the first step is similar to a normal analysis. However, the `tb_control` is set to be "summary" instead of "event". And besides of `mcd00900.hbk`, another output file, `trk00900.dat` is also generated with the summary of tracks.

At second step the `tb_control` is set to be "align" and `data_in_file` as `trk00900.dat`. As MINUIT fit is CPU consuming process, it is advised not to let all parameters floating at one time. The steering file `mn_skip.dat` controls which parameters are to be optimized, and the file is quite self-explained. The program can be in either batch mode or interactive mode, and the later is recommended. Try commands in MINUIT "showpar", "sim", "showpar" and "ret" for a test if you are not familiar with MINUIT commands. Then choose option 0 to quit. The output histograms are in file `aln00900.hbk`, or controlled by `hb_k_out_file`. It can be viewed with `manip.kumac` macro file. The optimized parameters are stored in `mn_geometry.dat` file in the same format as that of `tb_geometry.dat`. Due to that the beam tracks are parallel with very small dispersion in direction. The fit is not sensitive to Z locations and it is wise to determine Z with other methods. It is also recommended not to let more than 6 parameters floating especially when you are serious to use commands like "mino".

Monte Carlo generation

For beamtest 2000 a standalone MC was used to predict the resolution. The Kalman filter and other effects were not properly included. In this package the MC simulation is included and thus the simulation is more realistic.

To generate MC events, `tb_control` needs to be set to `mcgen`. Other things need to be set are `nevent` and `run_mc`. The output files are `mcd00900.dat` for raw data and `mcg00900.hbk` for histograms which can also be viewed with `manip.kumac`.

The user can also generate MC events including random trigger events as noise. The `tb_control` needs to be set to `mcran` and `data_in_file` points to the input file of random trigger events.

There are many parameters that can be tuned in MC simulation. For now they are not introduced.

Further analysis using CWN ntuple

The normal analysis run with `tb_control = "event"` provides standard selections. It generates a `CWN` ntuples for further study where detailed event selection, eta correction of hit locations and so on can be easily introduced.

User can use in-line selection and conversions in `PAW` or `ROOT`. It is recommended that you create a FORTRAN format function that you can select event, fill histograms and so on. An example of such function, `anal.f`, is in `testbeam/example` directory. An macro file, `anal_example.kumac`, which uses such function is also created. And you can run it in `PAW`.

To generated a template yourself, do the following.

```
paw
h/file 1 run01980.hbk 0
uwfunc 2 your_template.f
quite
```

The `CWN` ntuple `ID=2` are saved one entry per selected track. That is to say, one event can have more that one entry is there are more tracks selected in the event. It contains three data blocks: `TRACK`, `PLANES`, and `SIXY`.

The first block `TRACK` contains track level information as listed below.

index	Variable/ Array	Brief explanation
1	ievt	Event ID
2	ntrks	Number of rough tracks in the event.
3	n_klm_ok	Number of good Kalman fit track in the event.
4	itrk	The track number(index) of this entry in the event.
5	klm_ok	The Kalman status (0: pass).
6	x_int	The interception point and the slope of the track at the first plane(iz=1).
7	x_slp	
8	y_int	
9	y_slp	
10	chisq	The χ^2 of the fit with all hits included.
11	cl	The confidence level calculated using <code>prob(chisq, ndof)</code> .
12	ndof	Number of degree of freedom of the fit. Note that some Y information are not included in the fit due to Kalman filter.
13	fixyslp	Whether or not the Y slope of the track is fixed (1: yes, 0: not).
14	yclass	Yclass of the track that indicates the Y quality of the track, 1 and 2 are good.
15	nplane_hit	Number of planes that has hits included in the rough track.

The block `PLANE` contains cluster hit and Kalman filter information of the track on each plane. The variable `nplane` indicates the number of planes presented in the geometry files. So it is a constant. The rests in the block are arrays of `nplane` elements, each corresponds to one plane that is sorted in ascending `Z`.

index	Variable/ Array	Brief explanation
1	nplane	Number of planes in geometry file. It indicates the size of following arrays.
2	plane_id	Plane ID (ordered with ascending Z).
3	itype	Type of the detector(1:SPD_X, 2:SPD_Y, 3:SSD_X, 4:SSD_Y).
4	iort_row	Orientation of the row (-1: row <-> -X / -Y).
5	nclus	Number of clusters in this plane.
6	iclus	Index of this cluster, (0: not present).
7	npix	Number of pixels in the cluster.
8	nrow	Number of rows/columns the cluster covers.
9	ncol	
10	rowb	
11	colb	Average row/column with binary weight.
12	rowq	Average row/column with linear charge weighting.
13	colq	
14	eta	eta of charge distribution.
15	sumq	Sum of charge in Ke.
16	xg	MC generated interception point.
17	yg	
18	xl	Local X/Y.
19	yl	
20	xsig	Assigned resolution.
21	ysig	
22	xw	Hit point in world coordinates.
23	yw	
24	zw	
25	rxl	X residual with the hit included in the fit.
26	srxl	X error of the residual ($srxl^2 = xsig^2 - spxl^2$).
27	spxl	X error of the projection (xf).
28	ryl	Y residual with the hit included in the fit.
29	sryl	Y error of the residual.
30	spyl	Y error of the prediction.
31	rxo	X residual with the hit excluded in the fit.
32	srxo	X error of the residual ($srxo^2 = xsig^2 + spxo^2$).
33	spxo	X error of projection (on xf).
34	ryo	Y residual with the hit excluded in the fit.
35	sryo	Y error of the residual.
36	spyo	Y error of the projection.
37	xf	Interception point from the fit.
38	yf	
39	rhof	Correlation coefficient of the prediction error matrix. It is not useful in the default <code>strip_equivalent</code> fit.
40	xslp	Slope of the track at interception.
41	yslp	
42	infix	X/Y information included in the fit.
43	infity	

The last block, `SIXY` contains all raw hits in the event. The variable `nhit` indicates the number of all hits. The rests in the block are arrays of `nhit` elements, each corresponds to one hit.

index	Variable/ Array	Brief explanation
1	nhit	Total number of hits in the event.
2	ipln	Plane ID of the hit.
3	icol	Column/row ID of the hit.
4	irow	
5	iadc	ADC value.
6	icls	Cluster number (index) in the event that this hit belongs to.
7	qval	Charge value (Ke).

Histograms IDs

The histogram ID ranges for different actions and different are assigned as following. For different `tb_control` different histograms present. If the histogram is for each plane, the lowest two digits represents plane ID. The CWN ntuple ID is 2.

Base ID	Brief explanation
11100	geom parameters
11200	geom parameters after alignment
11300	global parameters
11300	detector parameters
11400	sensor parameter mapping for MC
11500	charge parameters for MC
21000	simulated track and hits
30000	track summary write out
31000	track summary reading
32000	MINUIT fit
41000	hits information
42000	cluster information
43000	position - mcgen
51000	form track
61000	Kalman filter

To find out which histogram corresponds to what information, user needs to either list histograms in `PAW`, or check the source code. However, the most convenient way is to use `manip.kumac` macro file in `PAW`. It views through most of the useful histograms.

Please send your comments to: jwang@physics.syr.edu